# Minerva : A Smart Video Assistant for the Kitchen

by

**Shyam Krishnamoorthy**

B.Tech., Computer Science and Engineering (1999)
Indian Institute of Technology, Bombay, India

---

**Submitted to the Program in Media Arts and Sciences**
**School of Architecture and Planning**
**in Partial Fulfillment of the Requirements for the Degree of**
**Master of Science in Media Arts and Sciences**

at the

Massachusetts Institute of Technology

September 2001



Signature of Author:

---

Program in Media Arts and Sciences
July 25, 2001

Certified by:

---

Dr. Andrew B. Lippman
Principal Research Scientist
Program in Media Arts and Sciences
Thesis Supervisor

Accepted by:

---

Dr. Andrew B. Lippman
Chairman, Departmental Committee on Graduate Students
Program in Media Arts and Sciences

Minerva : A Smart Video Assistant for the Kitchen

by

Shyam Krishnamoorthy

Submitted to the Program in Media Arts and Sciences
School of Architecture and Planning
on August 14, 2001 in Partial Fulfillment of the
Requirements for the Degree of Master of Science in
Media Arts and Sciences

## ABSTRACT

Minerva, a video assistant for cooking, is controlled by the actions of the user. Specifically, the content and format of the video shown on Minerva's display are decided by the food that one places on the kitchen counter. Thus, it aims to develop new ways of interactivity and control for media that solve the problem of content choice in an innovative and useful manner. In parallel, it is a study of the applicability of a generalized image-matching algorithm [1] [2] and into reflecting a program's awareness of the context of its execution in its behavior. Minerva is, for the user, a personalized kitchen assistant, a culinary knowledge-base and a cooking tutor, all built into it by design rather than by accident.

Minerva tackles problems faced by current computer systems, including trade-offs between automatic program execution and user control, access mechanisms for media from large databases, retrieval techniques for recipes and cooking information, and embedding computing systems in the home environment with useful features without compromising on non-obtrusiveness.

The system has a camera that looks at the kitchen counter and a recognition engine that analyses the ingredient placed under the camera. Together, they form part of the input that decides the choice for the content of the media displayed. Contextual variables, including the user's identity and preferences form the other part of this input. A database look-up program that returns the location of the video and recipe image appropriate for these input forms the retrieval sub-system and a player for the cooking video retrieved displays the media chosen. Automatic control for the media is enabled wherever possible, allowing for prior information about the user to affect the choice and format of the cooking video displayed.

This thesis describes the motivations, concept and implementation of Minerva, ending with an evaluation of its performance and impact, as well as directions for future work.

Thesis Supervisor: Dr. Andrew Lippman
Title: Senior Research Scientist, Program in Media Arts and Sciences

This work was supported by the Digital Life Consortium, MIT Media Laboratory, 2001

# Thesis Committee

**Supervisor**:

Dr. Andrew Lippman
Senior Research Scientist
MIT Media Arts and Sciences

**Reader**:

Dr. V. Michael Bove Jr.
Principal Research Scientist
MIT Media Laboratory

**Reader**:

Prof. Ted Selker
Associate Professor
MIT Media Arts and Sciences

## Acknowledgments

I have many people to whom I owe my happiness and the new opportunities I encountered during my two years at the Media Lab. I am grateful to all of them.

I would like to thank my advisor, Andy Lippman, for the guidance, opportunity and resources he has provided me during the course of my degree.

I am grateful to my readers, Mike Bove and Ted Selker, as well as Ms. Barbara Wheaton, for sitting down with me and helping me refine my ideas, and my thesis.

Thanks to Sailu and Bryan for making the project easy and fun to work on, as well as helping to put all the parts together in time.

Stefan Agamanolis and Nuno Vasconcelos have done great work in their respective fields, and I thank them for helping me understand and use that work effectively.

Kim Schneider, Deborah Widener and Polly Guggenheim have made our lives so easy that I do not know what the students would do without them.

During my two years here, my lab-mates Jim, Kwan, Ying, Thomaz and Floyd have been great friends, and bore me out wonderfully. Thank you.

Of course, I thank my house-mates and all my friends from school, IIT and MIT for all that they have offered me, and all that they have put up with. Its been a fun ride !

The three women in my life, Amma, Meera and Radhika have been my inspiration. Thank you for everything.

# Contents

# Chapter 1

# Introduction

## 1.1 System Overview

The Minerva system is a perception based cooking assistant for the kitchen. It consists of a camera that looks at the kitchen counter, and a touch-screen for the display. The user places food items, that he/she would like to cook with, on the counter and the camera takes a snapshot image of it. This image is passed to a recognition engine that identifies the ingredients. These ingredients are then matched with a database of recipes to find the dishes that best utilize these and match the user's preferences and profile. The user is then presented with a choice of recipes, from which he/she chooses one to watch. The system then plays a video of the cooking show, with customizations whenever possible. Preferences of the user, including his culinary and health choices, are taken into account and affect the choice of video to be shown, the manner in which the video is shown, as well as the initial screen where a television show is displayed (simulating the experience of a television in the kitchen).

Imagine the following scenario :

*Jeremy is thirty years old, a little overweight and interested in Asian food, especially seafood. He walks into his kitchen after work, turns his television on and opens the refrigerator. Sounds familiar? The illusion fades when Jeremy grabs some eggs and noodles, drops them on the counter and taps his television screen. The screen changes to show him three dishes he could cook with the eggs and noodles, placing the low-fat Asian seafood items on top. He taps on one of them and chooses a few more personalizations, and a cooking video for that dish starts playing, in the requested format – short and humorous. Jeremy is not an expert cook, and he was happy that all the dishes were easy to make, and used all his favorite ingredients. Not only were Jeremy's various food and health preferences taken into account, they even affected the way the video was*

*presented to him. Jeremy uses Minerva, not only for his daily food choice and cooking lesson, but also to figure what he can make with those spare items he has in his refrigerator.*

## 1.2 Motivation and Goals

Minerva is an exploration into the amount of interactivity that is necessary with computer video applications. Can a system function without any input from the user? Or is explicit input and complete control over the actions of the computer the way to go? One approach is to create systems that used as much information as possible from *what is already known about the user and the environment* to direct their actions, and thus use *minimal interaction* from the user. Minerva adopts this approach and thus allows the user to continue with the activity that he/she is doing, pausing only for minimal interaction with the system. This is well-suited for, and almost necessary when we try to embed computing systems in the home environment where the primary activity is not one of using the computer. Instead, the computing systems are inconspicuous and only serve to assist the person in his/her activity. The limited set of interactions that are chosen should be well-thought out, and placed in the system only if they are sufficiently important in assisting the task at hand. As mentioned in [9], it is necessary to take the context and environment in which the system will be used while designing the system.

Another aspect that shaped Minerva was looking at the applicability of a generalized image matching algorithm [1]. In Minerva, the same algorithm that is used to recognize the ingredients is also used as the base to split videos into individual shots and scenes [4], so that the viewer may directly jump to the next scene in the show if desired. Indeed, if the algorithm could successfully be adapted to both these uses, it could , with slight modifications, be applicable to many other uses, some of which could be face recognition, character recognition, web-based image searching and more.

Lastly, computing in the kitchen is a relatively new field. The Counter Intelligence group [12] at the Media Labs, MIT, builds computing systems that will be used in the "Kitchen of the Future". Using computers in the kitchen takes many forms – from enhanced micro-controllers in kitchen appliances, to new and innovative aids for existing cooking practices. Minerva is an attempt to use computing in the kitchen to simplify the process of deciding what to cook and learning how to cook it.

## 1.3 Usage Scenario

Imagine *Jeremy* of the previous example using Minerva for assistance in his everyday cooking activities. He is presented with various screens of Minerva, one for each step of the control flow of the program, and can switch between them based on what he wants to see. He starts by choosing the profile under which he will use the system. This process involves simply touching his name, displayed on the initial screen, on a touch-sensitive flat panel display. He would then be shown a television show, and it would automatically start with Jeremy's favorite channel.

The Minerva system consists of a main control process, written in the Isis programming language[16], that interfaces to the other modules of the program. Thus, this control module would keep track of the operating profile to ensure that Jeremy's preferences are taken into account while calling the various sub-modules.

When he is ready to start using the system, he presses a button on the screen and it changes to show the Minerva screens, Jeremy then starts using the system by taking food items that he would like to use in his cooking that day, and keeping them on the counter. At the press of a button on the screen, a camera looking at the kitchen counter takes a snapshot of the ingredients on it. This starts the recognition step.

The snapshot of the food items is passed to a recognition program [1] that analyses the image and identifies the ingredients in it. Multiple ingredients may be placed at a time. Based on the ingredients recognized, a recipe database is accessed and a query is done for the best matching recipe for the given ingredients. This query takes into account the dietary and health preferences of Jeremy. The query returns the location of the video of the cooking show for this recipe. Thus, if Jeremy had placed some chicken and noodles on the counter, Minerva will find recipes that use both these ingredients, as well as make sure that they are low-fat recipes. It uses information stored in Jeremy's profile to make these decisions. It then starts playing the cooking video for that dish.

Functionality to play, pause and stop the video, as well as to jump between scenes is provided. This functionality lets Jeremy continue cooking and watching the video in between. When he has finished boiling the noodles, if the video still shows the scene to cook the noodles, he can directly jump to the next scene and start learning how to cut the chicken.

A tag marks whether the video played is a normal video or a Viper video [17]. If it is a normal video, it is played on screen. If it is a viper video, Jeremy can then specify certain factors that will affect the presentation's length and content before it starts playing. So, if Jeremy is in a hurry or has already seen the video and wants a short version, only important scenes in the video will be displayed.

In summary, the camera and the touch-screen display act as the I/O components of the system, while the recognition engine and the database query program, together with the controlling Isis process, form the back-end components. A separate module [4] is executed prior to the system's use for splitting non-Viper cooking videos into individual shots and scenes. Together, these modules act under the control of the main module to create a smooth presentation and offer helpful assistance to users like Jeremy.

## 1.4 Overview of the Document

This thesis, including the introduction, consists of six chapters. The following chapter provides the background research that this work draws upon. The third and fourth chapters describe the design and implementation of the Minerva system. The fifth chapter looks at the performance of the system with a quantitative as well as a qualitative evaluation. The sixth chapter concludes this document, and looks at the lessons learned, improvements that can be made and future work in this direction.

# Chapter 2

# Background

This chapter steps through the background work that Minerva is based upon, and how each different piece of research ties in with the final implementation of Minerva.

## 2.1 Video, Interactivity and Media Control

Researchers have been looking at innovative ways of interactivity and control for video and other forms of media. Many projects using a variety of techniques have been explored. Some systems that led to the ideas behind Minerva are described below.

Christakos, Lippman et al. have looked at tagged representations of media – small devices that carry only a few bytes of data that identify the media stream that they represent [7] [6]. Once the required media stream is identified, it can be located anywhere on the local network or the Internet and accessed. Such a representation, using only an identification for the media stream, is more efficient as it removes the need to actually carry around the media on a compact disc or a DVD. The actual system in which this technique was used, a distributed network video player, was implemented with a multicast algorithm that was used to retrieve the data from the network. The interface consisted of small coin-sized tags that had a few bits of information in them about the stream they represented. When these tags were thrown onto a pad in front of the display, an RF tag-reader embedded in the pad would pick up these bits of information, use these to identify and access the requested data stream. It would then play the video on the display corresponding to that pad.

Another technique in the above project looked at visual and iconic representations of the media to be played. Thus, a poster of a movie would represent the video stream for that movie. Embedded in the poster was a light sensor that would detect the amount of light falling on it.

When the user wanted to play that particular movie, he/she would point a laser light from a pen-sized device at the poster. The poster would then transmit its identification and location on the network, and the closest monitor would start displaying the video. The same laser pens were also used for further controlling the video (start/stop/pause etc.).

Following this, Mueller, Thomaz and Lippman looked into controlling media directly through actions and physical objects. ImpactTV [15] simulates a remote control for television, in which the channel to be displayed is chosen by directly throwing relevant objects at the television screen. For example, throwing a child's toy at the screen changes the program displayed to a children's network show. Even though the technique is not directly deployable, as it does not scale to a large number of channels, it looks at a new way of control for media streams that may be well suited for application in other fields.

Arai, Machii et al. [5] have attempted to retrieve files based on the user's actions on real objects on a desktop. The user controls a light pen, and performs activities with the objects on his desk. A camera looks at this activity and retrieves text files indicated by the activity. These files are then projected back onto the desktop so that the user can view them.

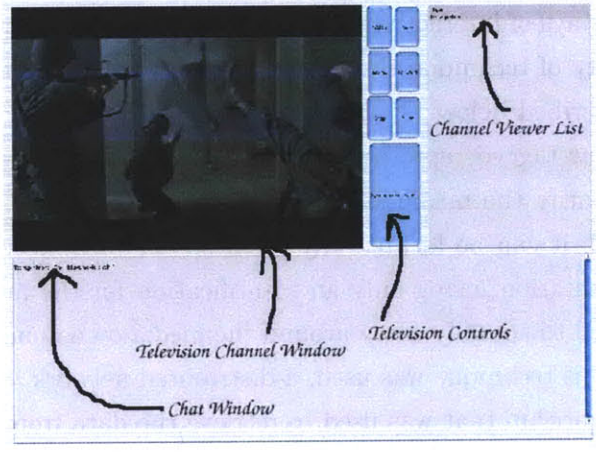



Figure 2.1: GroupTV

The author of this thesis, along with Lippman, has looked at interactivity and video when groups of people are involved. The driving force behind this research is that group interactions on a network scale exponentially, compared to polynomial scaling in one-to-one conversations. GroupTV made discussion groups that were linked to a particular television show. As people saw television, they would be able to talk to a set of people, all of whom were watching the same show that was playing. Flipping channels would thus change the "chat room" and the discussion group.

All the above projects were successful in creating a easy and intuitive interface for choosing the media stream. Minerva uses these ideas in its design. It looks at how interactivity and control in media could be placed in a situation where the interaction was implicit in the activity of the person, and this activity would be the control for the media stream. This approach

lends itself ideally to the home computing scenario where computing is not the primary activity and direct interaction is not always preferred. Thus, the user would continue in his activity of cooking, and the act of taking the ingredients out of the refrigerator and keeping them on the kitchen counter was the first part of his/her interaction. The general architecture of this system, viewed in this sense, is well applicable to other scenarios where users need to continue in their activity, without being distracted by the interface, but would still benefit from the information and behavior provided by the system. As an example, we could imagine the system being used in a gymnasium, with a program that understood the specific exercises followed by the user. This understanding could then drive the behavior of the system so that it could play a video of how to perform that specific exercise etc. Of course, as mentioned in the section 2.4, it would be important to make sure that such a system in sensitive to the context of its use. Thus, if the person is an experienced gymnasium user, it would be better to give him/her only high level tips and hints instead of a tutorial-style display and so on.

## 2.2 Visual Information Retrieval

One of the important components in Minerva is the recognition engine. This engine is based on previous work on image similarity measures by Vasconcelos and Lippman. This section gives a brief introduction to this technique. Even though other systems that identify foods have been more successful, Minerva utilizes this technique to explore the possibilities of using a generalized image similarity algorithm for specific applications. Specifically, the VeggieVision system at IBM [8] achieves a high rate of success at identifying over 150 different produce types. In spite of a relatively lower success rate using the system described in [1] [2], we see a significant contribution in the manner in which their algorithm can be adapted to many situations. A description of the considerations for image similarity taken into account in [1] follow.

The first step in searching a set of images by looking at their similarity is to characterize them sufficiently, so as to be able to distinguish between the categories. As it is impractical to directly compare the pixels, the image is first transformed to a feature space, and a feature representation is found for each image. This feature representation describes how the image populates the feature space. The search mechanism then involves finding an appropriate retrieval measure that uses this feature representation to analyze the similarity between the images.

**Image Representation** The author, in [1], shows that the feature transformation has to be invertible in order to decrease the probability of a wrong match. On the other hand, the feature transformation should fulfill the role of dimensionality reduction, so as to reduce the

complexity of representation in the feature space. Given these constraints, a trade-off between invertability and dimensionality reduction is sought in the choice of the feature transformation. This trade-off is found in the Discrete Cosine Transform (DCT), as it results in a relatively small number of relevant parameters and these have more perceptive significance than a direct Principal Components Analysis (PCA).

A feature representation is now sought that will describe how each image populates the DCT feature space. Traditionally, two approaches have been commonly used. Characterizing features by their moments, such as mean and covariance, has been used for texture recognition – the relatively homogeneous texture patches that in the standard Brodatz database are well approximated by a Gaussian density. Image histograms, the other common approach, have been applied to object recognition problems, and work well with the non-homogeneous images found in the Corel and Columbia databases. Unfortunately, the histogram technique has exponential complexity with respect to the dimension of the feature space and cannot be used effectively with all types of images. Vasconcelos therefore suggests using a Gaussian mixture model that is a weighted sum of individual Gaussian densities. They show that this model, when combined with Bayesian retrieval techniques, matches or outperforms other techniques for image similarity matching.

**Image Similarity** If we view the retrieval problem as one of Bayesian inference [3], the goal is to find a map from the set of feature vectors (described above) to the image class, that minimizes the probability of error. The author, in [1], shows that this technique can take advantages of prior statistical knowledge to better find the matching class for a given image. In Minerva, the system currently does not incorporate prior knowledge into the retrieval process, and all image classes are considered equally likely. In this case, the Bayesian retrieval technique simplifies to be a maximum likelihood (ML) classifier. This gives an augmented distance metric that performs more robustly than the Mahalanobis distance and returns a probability of error that tends to the Bayes classifier error, faster than the density estimates tend to the correct densities. This characteristic allows us to use coarser estimates of the density than in other algorithms, but with just as accurate results.

Once images have been represented in the feature space, a set of images belonging to the same class may be combined into a single representation using hierarchical mixture models, which combine many Gaussian mixtures into one mixture. This is also described in [1]. This technique is used by Minerva to make single mixture models for each image, which are then combined to form hierarchical mixture models for each ingredient. Minerva performs ingredient

recognition by creating a database with the hierarchical mixture models of training images of all the ingredients. After this, when a new ingredient is to be recognized, a mixture model of the image of the new ingredient is calculated. This model is then compared with each of the models for the training ingredients, and a probability measure representing this similarity is obtained using the above described technique. The ingredient of the training mixture models that is most similar to the mixture model for the query image is then taken as the matching ingredient.

## 2.3 Context-sensitive System Design

Minerva is situated in the kitchen – a busy, and at the same time personal, space in the home – and introduction of computing in such a space is bound to make the interface intimate and omnipresent. Such situations demand that the design of the computational system takes into account the environment and the people around it. The system should use this information to correspondingly modify its behavior, making it easier to use and more in touch with its surroundings. In [9], the authors describe the increasing importance of context-awareness in the design and implementation of a computational system.

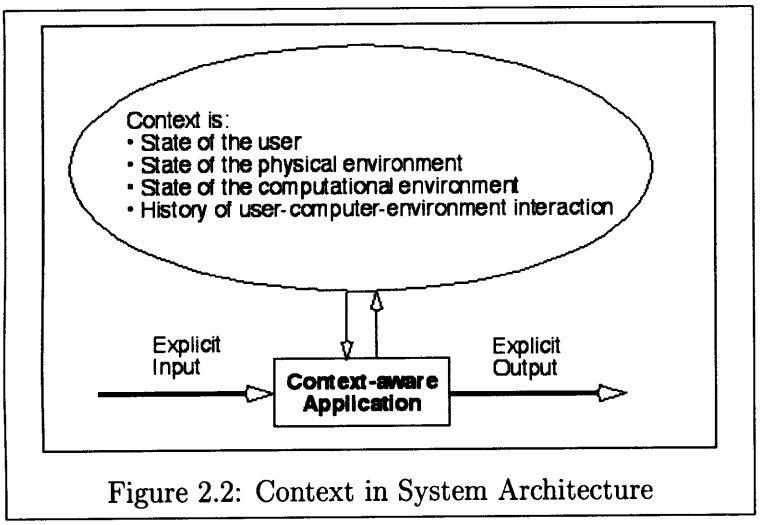


Figure 2.2: Context in System Architecture

Context-aware systems should sense and remember information about the person and the situation in order to reduce the communication and effort required on the part of the user. To effectively reach this goal, the system must have this information built into its creation and maintenance of user, task and system models. In addition, these models should drive the behavior of the system to effectively perform its function. The notion of context-awareness and the importance of the user, task and system models are further described in [10], and the diagram indicates the status that contextual variables occupy in a computer program. In addition, the authors argue that context in the case of user interface design should translate into its visual language. It is also important to simplify the interface without reducing the functionality. These factors are taken into account in the design of Minerva.

In the case of Minerva, the user model is the most important, and it is created on installation. The model contains information about the user's current state as well as his/her culinary and health preferences. This information is maintained in a database and can be updated or learned during the system's execution. The main control process takes into account the user model to drive the behavior throughout the use of Minerva. The user model not only controls the movie that is played on starting, but also the recipes that are chosen for a given set of ingredients, the order in which the recipes are displayed and the format in which the final video is played (if it is a Viper video [17]). The computer task model keeps track of the screen of the Minerva program that is currently active, and handles the behavior accordingly, while the system model keeps track of the system state variables (is the movie playing or stopped etc.).

## 2.4 Isis and Viper

Isis is a programming language for responsive media [16]. It is specially designed, in syntax and in internal structure, for fast and complete development of media applications and the interfaces for these applications. Using a core language element, Isis has multiple layers, each with additional functionality, both in the form of different abstraction levels, and in the form of software libraries. Isis supports video capture, OpenGL and SDL. Dozens of projects, in such diverse fields as aware portals, interactive art installations and computer vision have been able to exploit the advantages that Isis offers in terms of ease of use and functionality. All this makes it an ideal environment for creating the Minerva user interface as well as the main control module. Since Isis is an interpreted language, computation intensive processes like image processing are written in native C code and invoked by Isis. Minerva uses Isis for the main control loop, but calls pre-compiled C code for the image recognition, database look-up and object separation routines.

Viper [17], built on Isis, is a system that aids the creation of responsive video – video sequences that have multiple streams encoded in them instead of one, and from which a single sequence is organized to make a presentation. The content creator can decide how the individual clips are put together to create a complete video, in a simple but powerful manner, and even include various external factors that affect the final organization of clips in the video. Minerva is ideally suited to apply Viper to create and present cooking videos that are specifically tailored to the user's preferences. Contextual factors pertaining to the user and the environment (culinary preference, items in the refrigerator etc. ) can be used to create a video that is specific to the situation. As an example, if the user likes sea food and he/she is out of butter, the video shown could be a

version that makes the same dish using shrimp, and uses margarine instead of butter. A special Viper production was made for Minerva – a cooking show for the recipe "Thai Mango Curry". Creating such a Viper movie with any system involves the following steps:

- Planning: The creator of the productions needs to decide what the video would consist of, what variables affect the behavior of the program, how these variables will be collected and represented, how each of these factors will translate into control factors for individual clips, as well as how to make the production in a simple and efficient manner.

- Capturing Media: Since multiple clips, which will form different parts of the final version of the movie displayed, have to be shot, it is practical to think of ways to reduce the amount of time making the video by intelligent use of the capture process. For example, in the case of a cooking video, all the clips linked to a particular style of the dish could be shot in one go, and then they can be edited into individual clips. The media, after capture has to be digitized, currently into the Isis movie format. Various converters can be used for these, including special utility functions (isisu) that have been packaged with the Isis library for this purpose.

- Annotation: A utility program, provided with the Viper tools, allows the content creator to edit the video into a database of individual clips and annotate them with appropriate keywords, scale factors etc. Thus, each clip that will be part of some version of the final movie is extracted from the videos and annotated with keywords that describe that particular clip. These keywords could correspond to objective features of the clip like its chronological order, length etc. or subjective features like its importance, the level of humor in it, whether it is a violent clip etc.

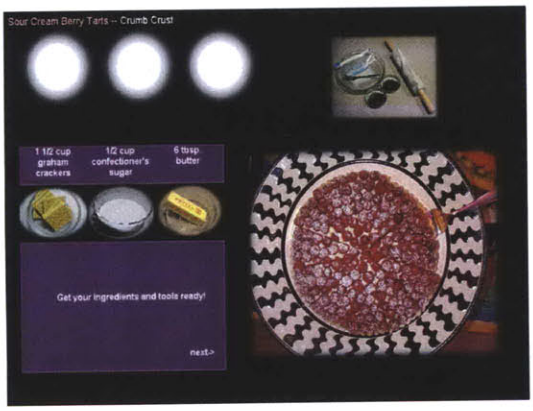
- Editing Guidelines: These are, in some sense, the most important pieces in the Viper production process. The editing guidelines lay out rules for putting together an Isis video from the individual clip database created earlier. The sections defines the relationships between the factors that affect the presentation, and the resultant expected values of the features annotated on the separate clips.

- Analysis: Once the above stages are completed, the final presentation can be viewed using tools that are provided with the Viper system. These tools allow the content creator to see the videos and their responses to the various factors that affect them. The tools also allow for graphic visualization of the sequencing of the individual clips that have been extracted from the database to create the final presentation.

Once the above steps are completed, the production is integrated into another software system (in this case, Minerva) by using Macaroni [16] objects that encapsulate the Viper videos. Functions are provided to create a presentation from inside any Isis program, and the factors that affect the video are simply passed as parameters to these functions. Convenience functions that play and pause the video are also provided, and they handle both the audio and video for the final presentations. In addition, the resultant video can be embedded into the rest of the interface that has been created using Macaroni.

## 2.5 Computing in the Kitchen, and for the Kitchen

**Computers in the Kitchen** When looking at the applications of computing in the home environment, it is impossible to ignore the effects of technology and computational devices in the kitchen. Kitchen appliances have more and more intelligence built into them, from smart refrigerators that keep track of the food inside them [13] to microwave ovens that know when the food inside gets cooked. Taking a different twist from the many "Houses of the Future" in research labs, the Counter Intelligence group [12] at the Media Labs, MIT, is creating the "Kitchen of the Future". There are various projects that look at the applicability of computers in the kitchen and the many useful and fun ways in which they can be deployed to aid in the activity of cooking.




A white paper [22] details the projects, past and present at the Counter Intelligence group at the Media Laboratory, MIT. It also gives inspiring looks at future projects in the field of kitchen computing, as well as some theories and concepts in the area.

One of the projects that tackles a subject close to Minerva is CounterActive by Ju et al [14]. CounterActive converts the kitchen counter into an interactive cookbook, and provides instructions and pictures like a regular cookbook, but can provide music, movies and help on touching appropriate hot-spots on the counter. It achieves these by using a projected display onto the table top and an array of magnetic field sensors under the table.

CounterActive is an example of a captivating way, for children and adults alike, to learn cooking using computers. Minerva is partly inspired by CounterActive as an example of building computing in the kitchen and creating a user-interface that is natural and unobtrusive in the home environment. Minerva aids in deciding what to cook, and indeed, could invoke Counteractive to teach people how to cook after they decide what to cook. Especially interesting would be the outcome of using a totally wired kitchen, such as those described in [23], along with Minerva. The system could then use the additional inputs from the refrigerator inventory to modify its presentations – or the instantaneous state of various devices could affect the behavior of the program. Indeed, Minerva has all the software components to support such a system already in place, and it is possible to have the computer pause the cooking show and inform you that your milk is boiling over etc.

**Computing for the Kitchen** Another way of looking at Minerva is as a search program for recipes. Recipes are necessary information sources for most cooks, and have traditionally been culled from various sources. The first printed cookbook, with a collection of recipes, was printed in Latin circa 1474. Since then, recipes have appeared in various places and forms.

Though cookbooks and magazines (e.g. [25]) are common sources of recipes, recipes are also found in advertising pamphlets and packing boxes from various commercial establishments that produce food or appliances and utensils for cooking. Since the advent of computers, programs have been developed that generate customized recipes, when factors like the number of servings required, the spiciness needed etc. are specified to the program.

Television cooking shows also started appearing in the early 1950ś (e.g. cooking shows by chef Dione Lucas) and provided cooking advice and helpful hints. The latest source for recipes, is of course the Internet, and many online sites have lots of recipes on them, organized by cuisine, ingredient etc.

Thus, any person needing a recipe or cooking advice has many sources that he/she can gather them from, but the missing component is a way of directly finding the recipe that they would like to use. Minerva aims to bridge this gap, and provide an intuitive and simple way of finding recipes that are appropriate to the situation.

## 2.6 Temporal Classification of Video Sequences

Challapalli and Lippman [4] have applied the image similarity techniques described in [1] to separating the temporal units in a video sequence. Using the similarity measures in [1], the authors compare subsequent frames in a video sequence to identify shots and scenes in the

sequence. A shot transition is identified when the image similarity measure between successive frames suddenly drops to a low value. A set of shots that belong to the same scene are also hierarchically modeled. This technology has been utilized to parse the cooking videos displayed by Minerva into individual scenes and the user can directly jump to the next scene or previous scene in the show if he/she so desires. This extra feature while viewing the video adds to the convenience that is necessary when the user has to interact during the process of cooking.

The steps for analyzing a video and splitting it into various shots and scenes are:

- Create feature representations for each frame image of the video, using the techniques described in [1].
- Find the similarity measures between consecutive frames each pair of frames throughout the video.
- Analyze these similarity measures to find troughs, where there is a low probability that the two consecutive frames are similar.
- These points indicate shot transitions.
- Create higher level mixture models (hierarchical) based on the density functions for all the frames in each shot. Perform the same step as above to determine scenes comprised of many shots, but form a single temporal unit in the sequence.

The above steps depend on cutoffs for the probability that are determined by a statistical analysis of the similarity measures found in the entire image. A research question to be solved is a generic technique that will calculate the appropriate cutoffs that will work with different kinds of video sequences. For example, if there is a slow dissolve or fade transition between two shots, the image similarity measures will not have peaks or troughs. Instead, they will change smoothly from high values to low values. Taking care of these situations currently involves testing the video sequence a few times with different cutoffs for the shot and scene separations.

# Chapter 3

# System Design and Architecture

This chapter describes the overall architecture of the system, as well as the design choices (and the rationale behind these choices) that were made while building the system.

## 3.1 Hardware and Programming Languages

Most of the hardware used for Minerva is standard computer equipment, but some of the functions required non-standard hardware and software. Minerva runs on a Redhat Linux 7 system with a 2.4.1 kernel. This system was chosen for its compatibility with the hardware to be used and the Isis programming language.

The camera overlooking the counter is an NTSC signal output camera and is connected to a WinTV Go card. The WinTV Go frame grabber PCI card, in conjunction with the bttv2 libraries [19] and the video4linux API[20] makes it convenient to grab still images and video from the camera. The Isis programming language has a module that handles the video capture and frame grabbing from the card. A Microtouch touch-screen monitor is used for the interface. A set of utilities were also developed by the author to drive the touch-screen functions under the Isis language.

The Isis programming language was also used to program the main control loop and the user interface. It was chosen for its ease of use and a complete set of features that allow the programmer to interface with all types of media input and output easily. The Macaroni module in Isis was used extensively to create the interface. The Viper tool for responsive video was used to create a responsive video production.

A miniSQL database [21] was used to store information about the recipes, ingredients and user preferences. It was chosen for its simplicity, lightweight nature and complete set of functions.

The recognition engine was written with a combination of C code and Unix shell scripts. The DAT library for storing data in an accessible way was recompiled on the linux platform to enable the recognition engine to function properly.

## 3.2 Interface Design

The over-arching goal in designing the interface was to make it as unobtrusive as possible, while retaining its functionality. This required it to have minimum interaction with the user, so that he/she may continue with their current activity. In this case, the activity is cooking, and it is necessary not only to let the user continue with his/her work, but also to give a smooth and continuous presentation at the same time, that will help the user in the cooking process.

**The Camera** One of the functions the program required to do was to get a list of the ingredients that the user wanted to cook with. Feasible solutions to achieve this were to use a camera to recognize the foodstuff, or to use radio frequency ID tags (RFid) that would mark the food containers with information about their ingredient content. The camera interface was chosen to recognize the ingredients for three reasons. Firstly, the camera can be placed in any place without intruding physically into the actual objects recognized. In other words, the RFid tags could not be placed on all food stuff (fruits, for example) directly. Secondly, it was also impractical to expect all food containers to be tagged using RFid. It is unlikely that all food sold from the market will be kept in containers that are tagged, nor is it practical to restrict users to such containers. Lastly, this gave us a way of exploring the ways in which the algorithm developed by Vasconcelos et al. could be used for a new application. A camera interface is not perfect at recognizing the ingredients, but it is so generic that it could be applied to other situations where the user's activity must be monitored. As in the case of the gymnasium example given earlier, a gesture recognition system could be adapted and used to extract information about the person's exercises and display appropriate information.

**The Touch-Screen Monitor** The program also required a means of getting input from the user about his/her choices, and a means of displaying the video and recipe. A touch-screen was an ideal solution as it was both an input and output device. As an input device, it is more suitable than voice recognition, both due its robustness, and also the fact that voice can get blanked out when the video is playing. Of course, the traditional mouse/keyboard interface is too cumbersome and intimidating for the kitchen environment. The particular touch-screen that was obtained also had a special shield that enabled it to function even when there was a small

amount of dirt on its sensing surface. This, though only a small additional feature to a monitor, is significant because it is a very likely situation that people have stuff on their fingers while cooking.

**Minerva Screens** Individual screens are displayed to the user at each stage of the Minerva interaction. Details and images of these screens are provided in the implementation section of this document. The interface was divided into screens for many reasons. Firstly, on a conceptual level, they break up the task model of the computer into its separate stages very easily and it is possible to know what routine the system is executing at every screen. Secondly, new screens can be added or deleted very easily. Every new screen would have to describe:

- The components (buttons, images, videos) that belong to that screen.
- The routines to create the above components.
- The actions that should be taken when a certain component is touched.
- The transitions to the other screens that can be changed to, from the current screen.

This structure allows for simplicity of the programming of new screens, and to understand the behavior of current screens. For example, after production of the Viper video (refer implementation section), a screen had to be added to display this Viper video, since it had different display techniques than a normal video. The above architecture made it really easy to add this screen without changing any of the code for the other screens, except at the transitions between the screens. More details of the transitions between the screens are discussed in the implementation section.

**Functionality** Along with the design of the screens, it was also necessary to take into account the features and functionality that had to be available in the final interface, so that the user could continue with his activity and still get useful information from the Minerva system. This gave rise to some features that were included in Minerva for ease-of-use and applicability. They are

- The initial movie: The initial screen, after choosing one's profile, simulates a television screen, and can indeed be used to display live television if appropriately setup. This design was chosen as making the system operate on the television would not only give it the added functionality of television, but would make it more approachable and easier to use.

The usual reactions that humans have towards a computing interface will be replaced by a more natural approach to television, due to higher familiarity, at least in a majority of the population.

- The real-time display of the video capture screen: This was necessary for giving a clear feedback to the user about what was visible to the computer and what it would see. Feedback from the computer to the user is a necessary component of any good user interface, especially if it is designed to be easy to use.

- The design of the cooking show screen: This screen has a lot of functionality that was deliberately included to make the system usable in the kitchen environment. Firstly, the user may not be interested in watching an entire cooking show to prepare a recipe, especially if he/she is an expert cook. Thus, an image of the text version of the recipe is also shown so that he/she may look at it and continue with their cooking. Similarly, stop and pause buttons are offered to the user. This would be very useful for the user to keep doing their activity, and watching snippets of the video as they feel. Lastly, the buttons to jump directly to the previous or next scene – super rewind and super fast-forward buttons in a way – will make the user have the choice of skipping a scene that he/she has already done the cooking for, is not interested in, or has already seen. Similarly, they could jump back to the beginning of a scene that they might want to watch again. More details of the buttons and their functionality is described in the following sections.

## 3.3 Back-end Design

The Minerva back-end software consists of various components tied together by the control module. The design criteria for the components follows.

**Database** It was necessary at the outset that Minerva, to be a successful system, had to be able to handle thousands of ingredients and recipes to be successful. Ideally, the information about the recipes would be culled from various sources on the Internet. Such a system is not currently in place, though some researchers, including some at the Counter Intelligence group at the Media Labs, MIT, are working towards creating large databases of food and food history. Minerva, given this lack of a complete database, was to be able to handle its own database that had to be scalable. Thus, a robust, but lightweight system for storing the recipes and the ingredients that went into them was required. The MiniSQL system was chosen and installed as it satisfied these criteria while having a complete SQL vocabulary for the required operations,

and simple usage rules. The miniSQL program also had a C programming language interface to the server. Thus, C routines could be written for specific queries and then encapsulated into Isis functions that were called by main program in Minerva.

The MiniSQL database was also used for storing the user preference data. Thus a central system was used for retrieving any information about the user necessary for controlling the behavior of the program. The information in the database was stored in the forms of tables pertaining to user and recipe information. These tables formed a relational database, that were cross-linked. So, multiple tables could be accessed and their keys compared to gather relevant information.

**Object Recognition Module** Once the decision was made that the ingredients were to be recognized using a camera, and that the system created by Vasconcelos et al. would be used to achieve this, a good training and retrieval system had to be developed for the ingredients. The training system was designed to be as simple to use as possible, for future additions to the project. To this end, shell scripts encapsulating the training process were written and documented. The training process involves the following stages:

- Collection of images of food ingredient to be trained.
- Splitting the images into separate food items.
- Creating Gaussian density mixtures of the split images.
- Combining all the density mixtures into a Hierarchical Gaussian Mixture, that represents the entire ingredient to be trained.

The recognition process consists of the following stages:

- Take a snapshot of the ingredients to be recognized.
- Split the images into separate items.
- Create Gaussian density mixtures of each item.
- Compare the density mixture of each item to each of the trained ingredients and find the closest matches. This returns the set of ingredients that are identified to be on the counter.

These steps were implemented in C code, and shell scripts that stepped through the entire process were interfaced with Isis, so that they could be integrated with the main control module.

**Object Separation** When multiple ingredients are placed on the table, they have to be analysed independently by the object recognition algorithm. Thus, one must first separate the objects into multiple images and analyse each one separately. This separation algorithm was written in C, and is called by an Isis wrapper around the program. The algorithm works by scanning the image from the left to right and top to bottom. During the scan, it compares the image to an image of the table top when it is blank (taken during the initial setup of the system). Whenever an object is encountered (the color values are different from those of the background), it recursively goes through the object identifying its boundaries.

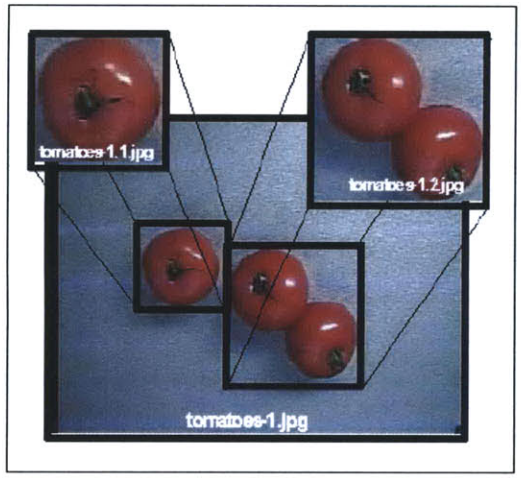



Figure 3.1: Object Separation

This recursive technique is similar to the fast floodfill algorithm described in [18]. Once the object boundary is identified, it is extracted and copied into a new image. Thus, the image with multiple ingredients is broken into individual images with one image each. One drawback of this algorithm is that shadows, objects very close to each other, or dirt on the counter cause multiple objects to be joined into one. The above image describes the working of the algorithm, as well as shows the drawback described above causing two tomatoes to be grouped into one image.

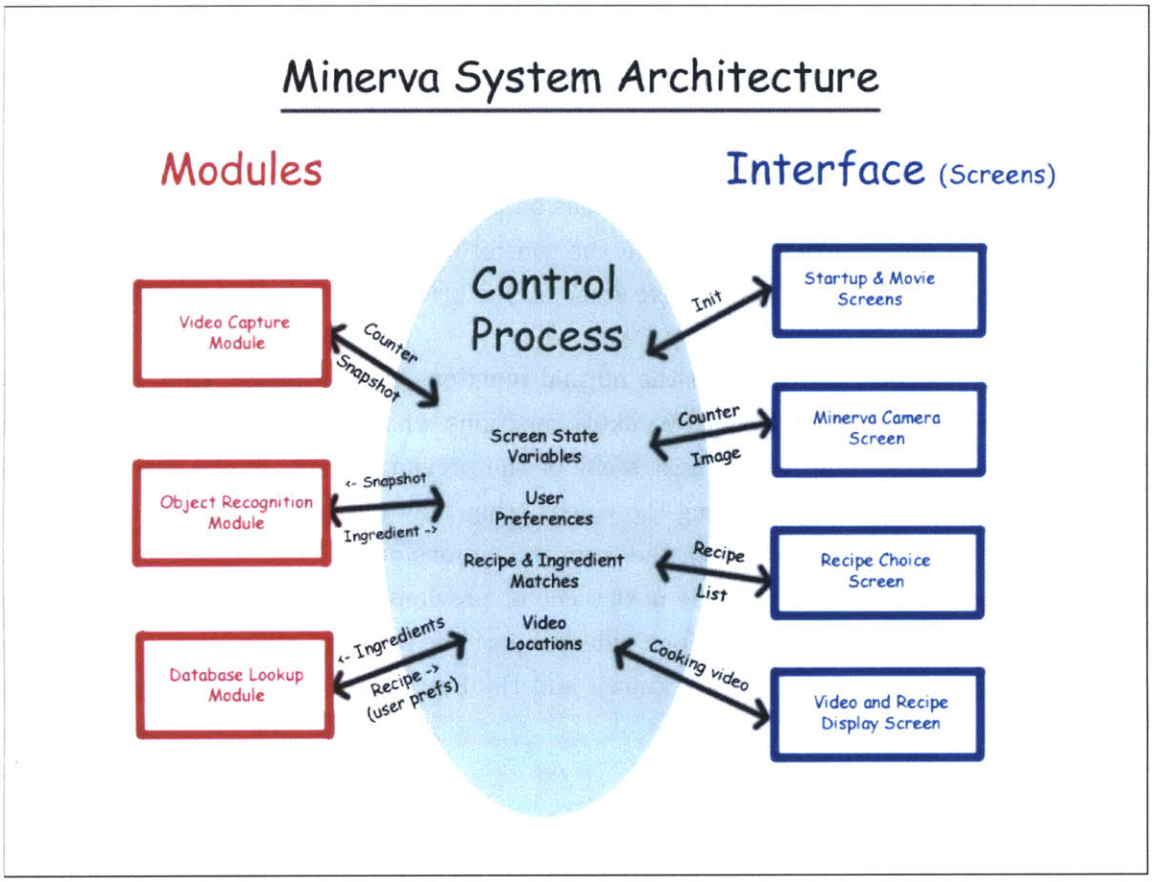
## 3.4 Control Flow

This section describes the overall architecture and control flow in the Minerva system. Figure 3.2 shows the interactions between the main control process, the modules and the interface.

The control flow in a typical use of Minerva happens in the following steps:

- The control module (CM) displays the initialization screens (displaying a profile choice screen and then a movie screen in the current implementation), waiting to go into the video capture screen.
- When the user chooses to look at the counter camera, the CM invokes the video capture process, changes to the Minerva camera screen and continually updates the screen with the snapshot of the counter top.

Figure 3.2: Control flow and Architecture of the Minerva System




- When the user places the ingredients on the table, he/she touches a button on the screen, giving the signal for the recognition phase.
- The CM receives a snapshot of the counter from the video capture module and passes this image to the object recognition module. This module performs the object recognition phase and returns the names of the ingredients identified.
- The CM passes the names of the ingredients and the user name from the previous phase to a database look-up program that finds the best matching recipe for a given set of ingredients and the user's preferences. It achieves this by retrieving the user's preferences, and searching for these preferences and the ingredients on the table together among all the recipes, assigning weights for each factor that matches. The highest-weight matching recipe is then selected. In this implementation, the top three matches are returned.

- Given the recipe names and the location of the videos, the CM changes the screen to let the user choose one of the matching recipes.
- After this choice, the video of the selected recipe and its corresponding textual version are displayed on the screen.

The control module also handles all the interactions on pressing the other buttons on the screen. Thus, in the section for a particular screen in the control module, the actions to be performed when each button (like forward/rewind/next scene etc) is pressed are specified in that section.

**Jumping Across Scenes** Other than the normal functions of play, pause, stop, rewind and fast-forward, Minerva also implements two extra functions while the video is playing. These allow the user to directly jump to the next scene or the previous scene. In the case of normal videos, this is achieved by first processing the videos using the technique mentioned in [4]. This outputs a list of frame numbers at which the scene transitions occur. The control module then uses these frame numbers to jump to the next scene or previous scene when the corresponding buttons are touched. In the case of Viper videos, since the video is already sequenced from smaller clips, the position of each clip is known and the buttons are used in the same way to jump between them.

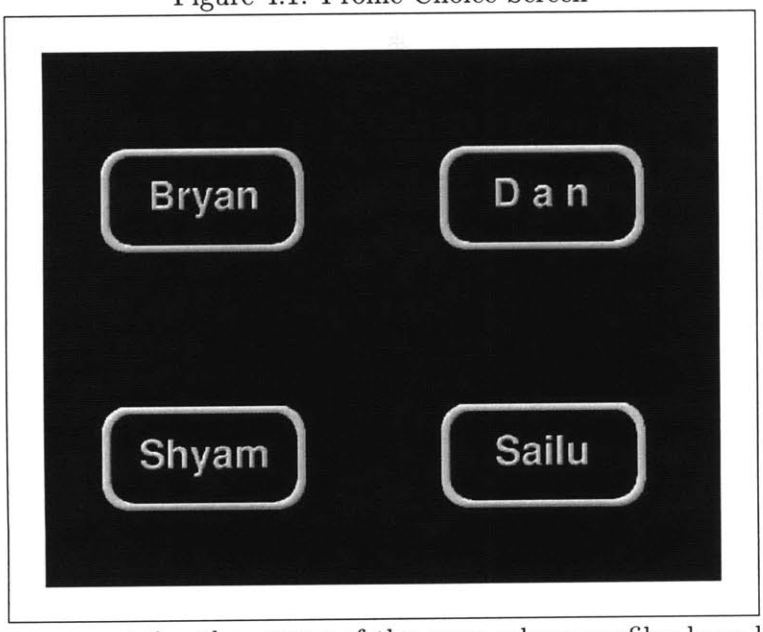
# Chapter 4

# System Implementation

## 4.1 Running Minerva

The following screens step through the execution of the current implementation of Minerva.

Figure 4.1: Profile Choice Screen




The opening screen contains the names of the users whose profiles have been stored in the database. Currently, this format for display looks good under the assumption that there are a small number (4-6) people in the household who use the system. For accommodating more people, a different form of display that is more scalable may be desired. Once the user chooses

the operating profile by touching one of the names, the name of the user is stored, and this will be used as the reference point for all database look-ups on the user's preference.

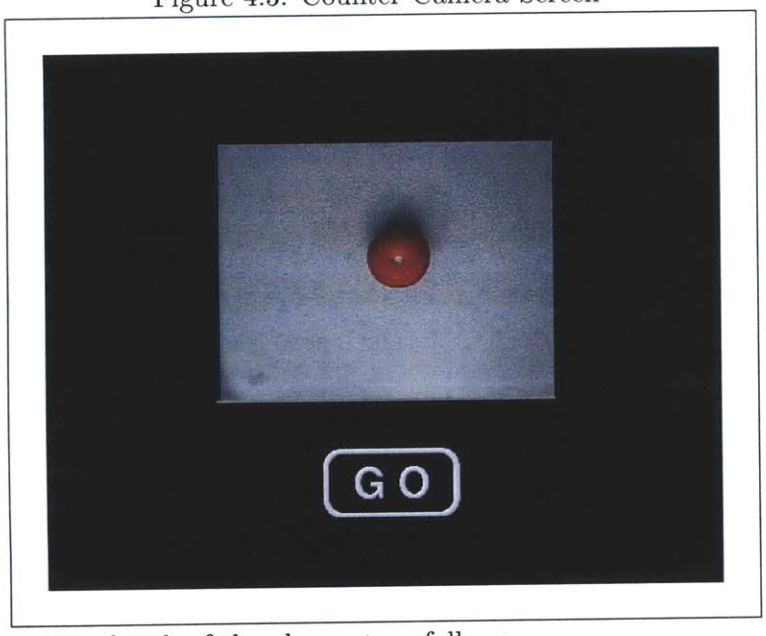
The program moves to the second screen, where a movie is displayed. The second screen is designed to simulate the experience of the now ubiquitous kitchen television. A favorite channel for each user is stored in the system, and this channel is initially started. Currently, the channels played are movies stored on the network, but they could be adapted to play live streaming video. Buttons for starting and stopping the video, as well as changing channels are provided. The movies are played using Isis movie objects, that are converted to Macaroni objects, which are rendered on the screen. The channel changing buttons, designated by a "+" and a "-" are implemented by circularly stepping through a list of filenames that represent the list of channels. If the channels are video played over the network, the filenames simply have to be changed to URIs (Uniform Resource Identifiers).

Figure 4.2: Initial Movie Screen


The "Minerva" button on the corner allows the user to go into the video capture screen, to start the recognition. This enters the main section of the Minerva system. Once the Minerva button is touched, the program initiates video capture from the camera and displays the image of the counter on the screen. The user can then place the ingredients he/she would like to use under the camera, make sure that they are seen by the camera by comparing with the image displayed on the monitor, and press the "GO" button. This causes a snapshot of the counter to be taken and passed to the recognition module. The recognition module returns a list of

ingredients matched. Then, a database look-up based on (a) these ingredients and (b) the profile chosen during the first screen, returns a list of recipes. This list is ordered according the weights they received during the database query – the weight depending upon the importance of the ingredients in the recipe and how well the recipe matches the user's preferences.

Figure 4.3: Counter Camera Screen




The implementation of each of the above steps follows:

- Video Capture: The video capture is done using the utilities in Isis. The routines used allow getting image snapshots from a frame grabber card that is supported. Currently, cards based on the Brooktree (bt8x8) chipset are supported, and are used along with the Video4Linux 2 API [20]. Frames are read continuously from the card as images, converted to Macaroni objects and rendered on the screen. Thus, the user can make sure that the ingredients are placed in a way so that they are visible to the camera.

- Recognition Module: When the "GO" button is pressed, the main program takes the latest image from the camera, and reduces its size by half. This is done using the image processing library in Isis. After this, the image separation technique described earlier is used to split the ingredients into individual images. For each of these images, a Minerva module performs the recognition and returns a list of ingredients. The recognition takes place by starting with conversion of the image into feature space. This creates .LIB files with the feature density representation of the image. This is then compared to the hierarchical feature

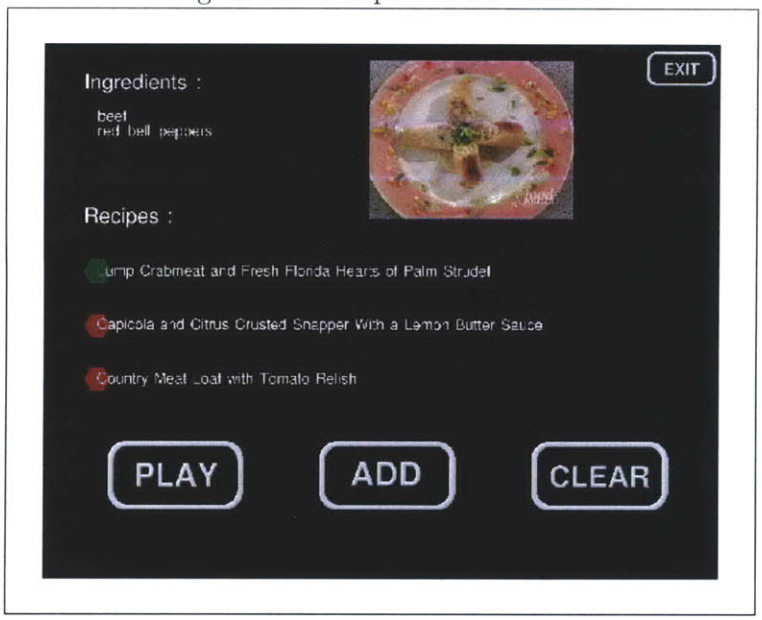
models created for each trained ingredient. This returns a list of probabilities of the query image being similar to each of the ingredients. This list is sorted and the top ingredient is accepted as the match. The name of this ingredient is written out into a file by the invoked shell script that does all the recognition steps (queryHierImage). The Isis function waits for the recognition to get over, and reads the file to know the name of the ingredient.

- The next step involves getting a list of three recipes that match this ingredient. This is done by invoking a database query program written using the C interface to the miniSQL server. The basic steps for the query are given below. Details about references to msql tables can be found in section 4.4.

  1. Open connection to msql server.
  2. Perform query to get all the preferences of the user whose name is stored in the profile. This is obtained from the user_preferences table.
  3. Perform query on recipes table and retrieve all the recipe IDs.
  4. For each recipe $R_i$
     (a) For each ingredient recognized, $I_j$
        i. If recipe $R_i$ contains ingredient $I_j$, add that ingredient's weight in the recipe to the total weight for that recipe $W_i$. This information is obtained using a query on the ingredients table.
     (b) For each user preference, $U_k$
        i. If recipe $R_i$ contains user preference $U_k$, add prefWT to the total weight for that recipe $W_i$ (prefWT is previously set to an appropriate value that is suitable for most recipes). This is obtained using the recipe_categories table.
  5. Find the top three weights amongst all $R_i$, by sorting and choosing the first three.
  6. For each of these three recipes, perform query to find their matching video file names, and expanded recipe names from the recipes table.
  7. Write these out to a specific filename.

- In the next step, the Isis program reads the file written by the query step and changes to the next screen.

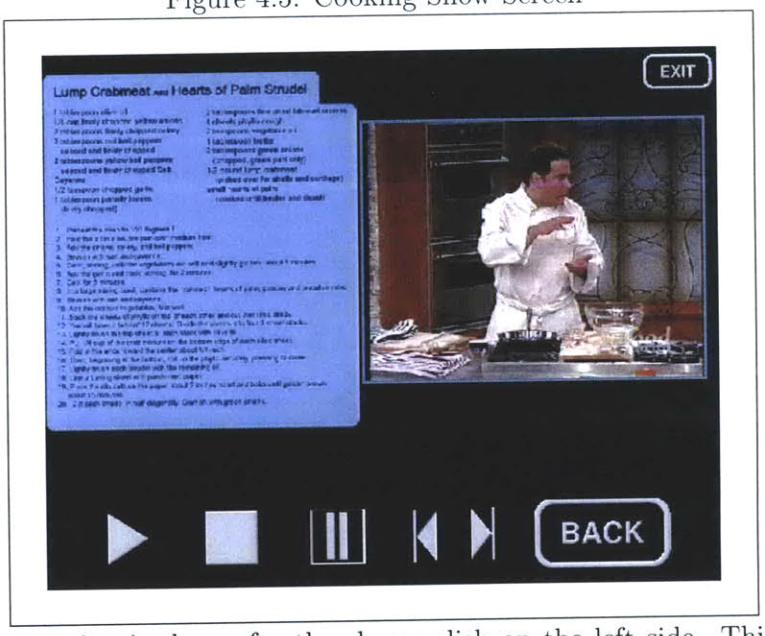
Figure 4.4: Recipe Choice Screen




The screen shows the list of the top three recipes, the chosen one being displayed with a green symbol. The "ADD" and "CLEAR" buttons allow the user to add more ingredients to the query or clear the list of ingredients shown to the system. This will return the user to the previous video capture screen. If the user chose the "ADD" button, the already recognized set of ingredients will be retained, and if the "CLEAR" button was pressed, the list will be made empty. The next set of ingredients recognized will therefore add to the list, or make a new set of ingredients, respectively.

On choosing a particular recipe, an image of the final version of the dish is displayed in one section of the screen. This is possible, since each recipe has an image associated with it, just like it has a video associated with it. Once the user chooses the dish he/she would like to make, the "PLAY" button causes a change to the next screen. If the movie is a Viper movie, more options are shown, and this is explained in section 4.5.

Figure 4.5: Cooking Show Screen




In this screen, a recipe is shown for the chosen dish on the left side. This recipe is shown from an image associated with it. On the right, a video of a cooking show for that dish is played on the right. Controls for the video include play, pause and stop, as well as buttons to jump directly to the previous scene and the next. The "BACK" button takes the user back to the recipe choice screen. The play, pause and stop functionalities are implemented using timers, which are provided in an Isis library. When the pause button is pressed, a timer value is saved, and when the pause button is pressed again, the timer is reset to the saved value. This timer is directly linked to the playing of the video, because the frame of the video displayed is computed from the timer value. The stop and play buttons are implemented in a similar fashion. The forward-scene and backward-scene buttons are implemented differently for Viper movies and normal movies.

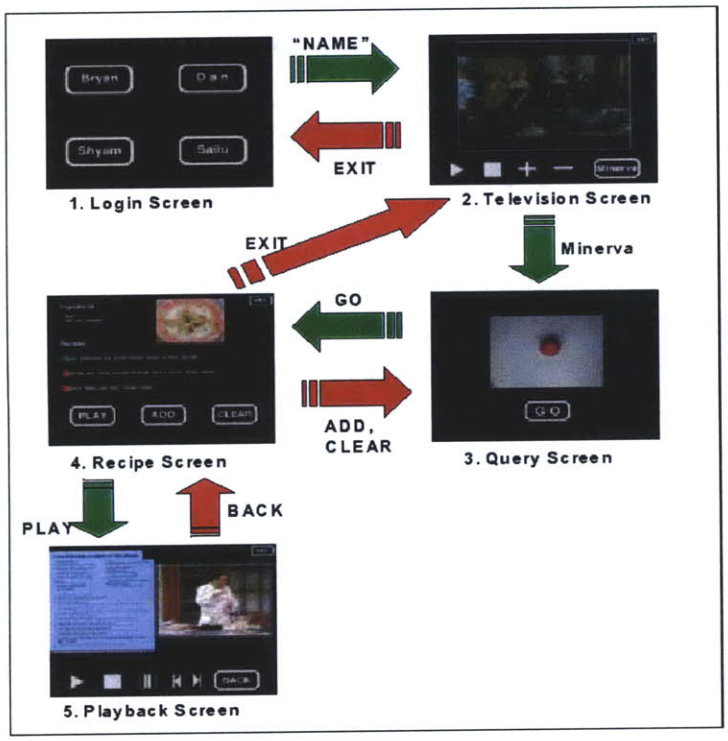
For normal movies, the algorithm described in [4] is initially run on the video files. This generates a new file with a list of frame numbers at which the shot changes happen. This file is read and stored during running Minerva. When one of these buttons is pressed, the current frame is compared to the list of frames to find that frame that is immediately before or after it. The video is then changed to that frame. The timer that controls the play of the movie is also correspondingly adjusted for the change in frame numbers.

For Viper movies, the sequence of clips that make up the presentation can be accessed from inside the program. Along with this sequence information, the start frame of each clip is available, and from this information, one can directly jump to the previous or next clip in the sequence.

## 4.2 Interface Screens

As mentioned in the system design and seen above in the execution, the display is organized into screens that have specific functionalities. Each screen has a set of transitions to other screens, and actions to be performed when buttons are pressed. The organization of the screens in the implementation is shown below. The arrows indicate the transitions that occur between the screens, as well as the buttons that trigger these transition events.

Figure 4.6: Minerva Screen Transitions



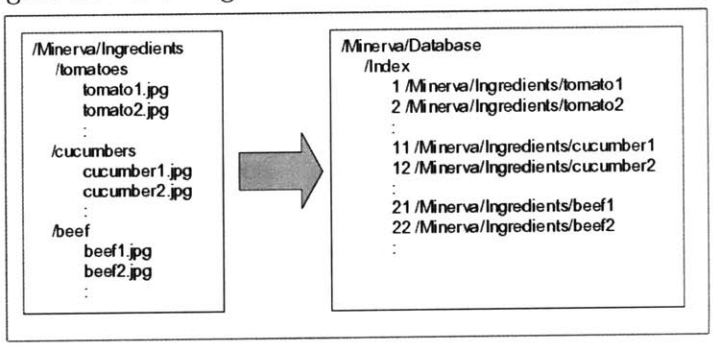

## 4.3 Recognition Module

The recognition module consists of two stages, training and identification. Both of these are based on the Gaussian mixture models, developed in [1].

**Ingredient Training** The training sequence was encapsulated into a Unix shell script. The system designer starts up the script and shows ten to fifteen versions of the ingredient to be

trained, pressing space when the ingredient is under the camera. An Isis function interfaces with the video capture from the camera and stores single pictures of the images. Another shell script creates mixture models of these images and stores them in a separate directory, along with an index of all the images pertaining to this specific ingredient.

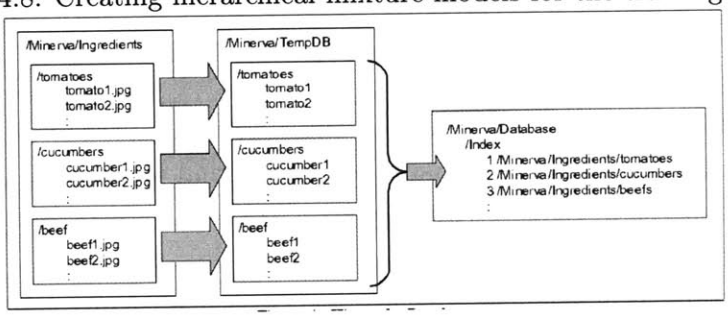
Figure 4.7: Creating mixture models for the training images




A shell script is then called that combines all these files, to create a common index of all the mixture models. Thus, when a query image is obtained, its mixture model is compared with the mixture models of each image of each ingredient and the ingredient of best matching image is returned.

An alternative to the above approach is to combine all the mixture models of an ingredient into a single hierarchical mixture. This creates an index file that has only one entry for each ingredient. When the query image is received, its mixture model is compared to the hierarchical model of each ingredient and the matching ingredient is returned. Both the above techniques were tried, and the intuition that the hierarchical technique is more robust was backed up by results from tests (given in the evaluation section).

Figure 4.8: Creating hierarchical mixture models for the training images



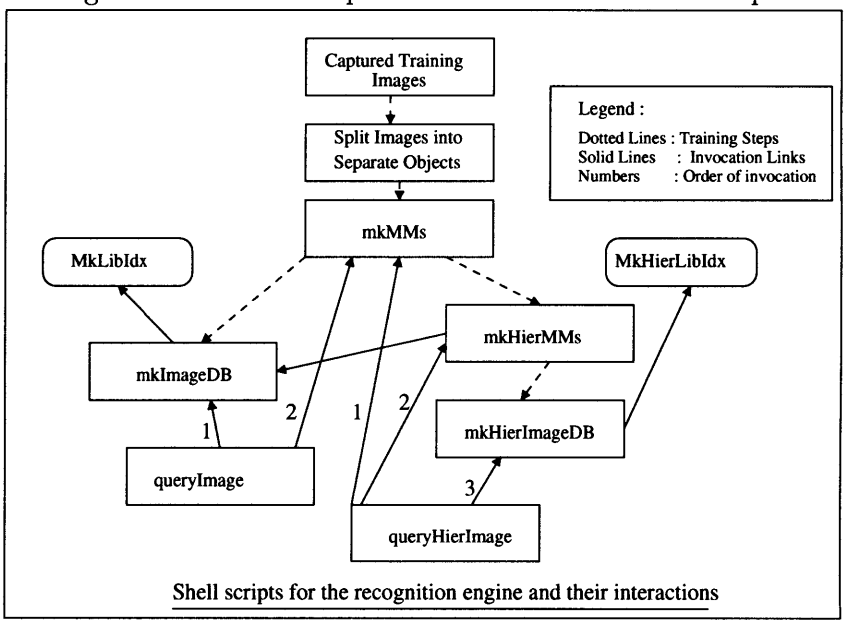
A complete description of the shell scripts created for use by Minerva follows:

- mkMMs: This creates mixture models for a set of jpg images – it creates .lib directories with the mixtures of each image.
- mkImageDB: This collects the mixtures of a set of images and stores them in a single database directory, in the form of .LIB directories. It also creates an Index file which has a list of all the images whose mixture models have been stored in the database.
- MkLibIdx: This is a small script that creates the Index file for the database. It is invoked by mkImageDB.
- mkHierMMs: This creates a hierarchical mixture model for the images. It first calls mkImageDB to create the database and then runs "hem", a C program on the databases to create hierarchical models of all the images in each database.
- mkHierImageDB: After hierarchical models are made which combine the models for all the images in each database, this script collects all these into a single database and creates a corresponding index file.
- mkHierImageDB: This is a helper script to create the index file. It is invoked by mkHierImageDB.
- queryImage: This script finds the similarity measures and returns the best match for a given image and a given training image database.
- queryHierImage: This script has the same function as the one above, but uses hierarchical mixture models for the query.

Minerva uses hierarchical mixture models during its operation. Thus, the training stage consists of calling "mkMMs <imgdir 1> <imgdir 2> ..." on all the training images captured, where each ingredient has all its images in one directory. . This is followed by running "mkHierMMs <imagedir 1> <imagedir 2> ..." on each of the above directories. Lastly, mkHierImageDB creates the training image database and index file. The query stage is simply done using a call to "queryHierImage <queryimage> <training image DB> <results file>".

The relations between the various shell scripts is indicated in the following diagram. The left branch indicates non-hierarchical techniques while the right branch gives methods that use hierarchical mixture models. It may be noticed that the mkHierMMs script (in the right branch) all invokes mkImageDB (in the left branch) during its operation.

Figure 4.9: Relationships between the various shell scripts




## 4.4 Database Tables

The database consists of information about the users and their preferences, as well as about the recipes, ingredients, the relative weight of each ingredient in a recipe etc. The format of the tables is given below:

user_info table:

| userID | login | first_name | last_name | age | experience |
|---|---|---|---|---|---|
| [int | [char] | [char] | [char] | [int] | [int 1-5] |
| 2 | shyam | Shyam | Krishnamoorthy | 23 | 3 |
| ... | ... | ... | ... | ... | ... |

The category field from the user preferences table is retrieved to control what recipes are fetched for a given set of ingredients.

user_preferences table:

| userID | category |
|---|---|
| [int] | [char] |
| 2 | vegetarian |
| ... | ... |

The next set of tables involve the recipes and the categories they fall into. When retrieving recipes based on a given set of ingredients, the importance of those ingredients in each recipe is taken into account in addition to the match between the categories of the recipe and the preferred categories for the user.

recipes table:

| recipeID | episodeID | recipe_name | filename | recipe_type | ingredient_count | difficulty |
|---|---|---|---|---|---|---|
| [int] | [int] | [char] | [char] | [char] | [int] | [int 1-5] |
| 1 | 2 | Mamas Macaroni | mac_cheese | Null | 6 | 1 |
| ... | ... | ... | ... | ... | ... | ... |

recipe_categories table:

| recipeID | category |
|---|---|
| [int] | [char] |
| 1 | vegan |
| ... | ... |

ingredients table:

| recipeID | ingredient | ingredient_wt |
|---|---|---|
| [int] | [char] | [int 1-10] |
| 1 | macaroni | 10 |
| ... | ... | ... |

The last two tables pertain to information about the videos that might be useful for cataloging and searching purposes.

shows table:

| showID | show_name | chef |
|---|---|---|
| [int] | [char] | [char] |
| 1 | Emeril Live | Emeril Lagasse |
| ... | ... | ... |

episodes table:

| episodeID | showID | episode_date | episode_title | notes |
|---|---|---|---|---|
| [int] | [int] | [date] | [char] | [char] |
| 1 | 1 | 4-Feb-2001 | Emeril's Restaurant, Orlando | Null |
| ... | ... | ... | ... | ... |

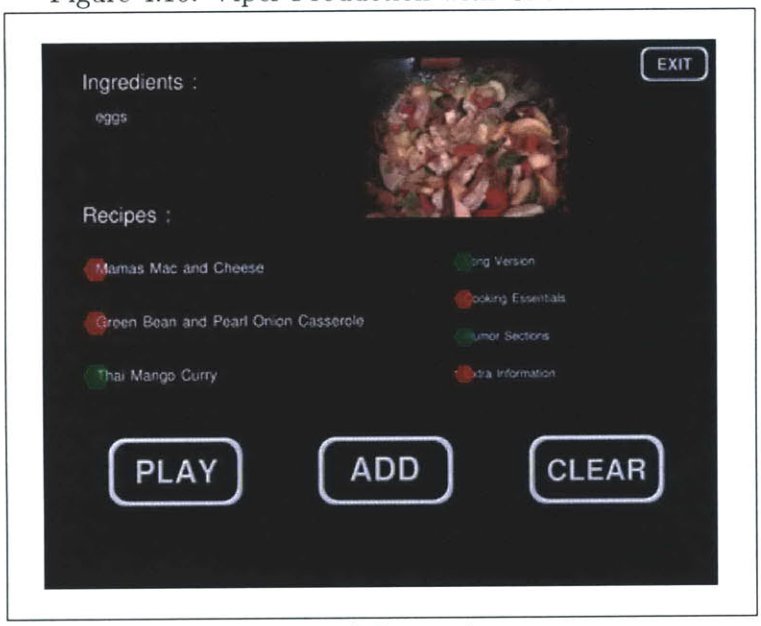
## 4.5 Use of Isis and Viper

The main control module is written in the Isis programming language, and all the externally called C programs (object recognition, database look-up etc.) have corresponding Isis wrapper functions that can be used. Also, since there were problems with the linux touch-screen drivers for the Microtouch monitor, Isis driver routines were written that directly communicated with the Microtouch hardware. These routines include resetting the monitor, waiting for a touch, waiting for a liftoff, and returning the X, Y coordinates of the position touched on the monitor. Macaroni, an Isis library for handling complex two-dimensional compositions of images, video and geometric shapes was used extensively for the interface. The library includes support for rendering images, video, lines, polygons etc. in real-time.

**Viper Cooking Video Production** A special cooking show, on Thai Mango Curry, was filmed for Minerva. It was a production that highlighted the ability of Viper to handle multiple versions of the same show, and Minerva using this ability effectively to tailor the presentation based on each user's culinary preferences. The same dish was constructed in three different ways. The first was a vegetarian version using tofu, the second was a seafood version using shrimp and the last was a meat version using chicken. The Viper system was embedded into Minerva and if the user chose to see that show, extra factors like length, humor, information level could be

set by the user, while the factor affecting the ingredient used would be extracted from the user's profile. Thus, the user had a cooking show that was made to his/her choice. The following screen shows the choices (on the right) that are offered to the user when the user chooses to see that program. In addition, the vegetarian or seafood version would be played to the user according to his/her profile information.

Figure 4.10: Viper Production with Choices for User




The factors that affect the organization of the movie are:

- Food style preference (vegetarian/seafood/French etc.)
- Length of movie
- Cooking Essentials
- Humor
- Additional useful information, such as health tips, food history etc.

Based on these factors, the clips were annotated with the following keywords and scales:

- time: This was used to organize the clips chronologically (a scale from 0 to 10000).
- importance: Important clips were those that were essential to the show and would be included in a short version. The unimportant clips would be dropped in the short version, but included in the long version (a scale from 1 to 10).

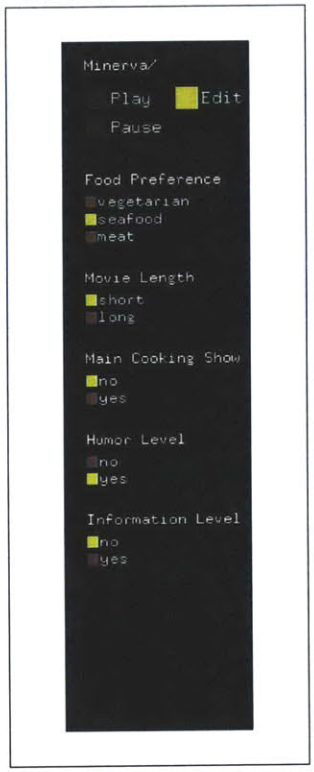
- common: Those clips that were marked common were common to all three variations of the dish.
- food-Style: This keyword marked those clips that belonged to only one of the variations (e.g. tofu specific clips or shrimp specific clips).
- main-show: These marked the clips that showed the essentials of cooking the dish.
- humor: Annotation for humorous clips.
- information: Clips that have extra information.

Editing guidelines were then written in the Isis language, which are processed by the Viper subroutines when a new presentation has to be generated using a given set of values for the parameters. The following rules were coded into the editing guidelines:

- The set of clips is split into those that are common, and those that are specific to each variation in food style. Only those clips that match the food-styles in the user's preference are taken into account.
- From the common clips, a smaller list of clips is obtained, by choosing only those that have a matching annotation with the user's choices from humor, information and cooking essentials (main-show keyword).
- The above process is repeated for the clips that show a variation in food-style.
- These two shortened lists are then merged into one large list of clips.
- From this merged list, another list is made using the "importance" keyword. If a long movie is requested, all clips are chosen, but only those with the "importance" scale higher than five (out of a maximum ten) are selected if a short movie is requested.
- This list of clips is then sorted according to the "time" scale to order it chronologically.
- This final set of clips is sequenced to form a movie. Currently, no effects like fades and dissolves are used, but Viper provides functionality for such features.

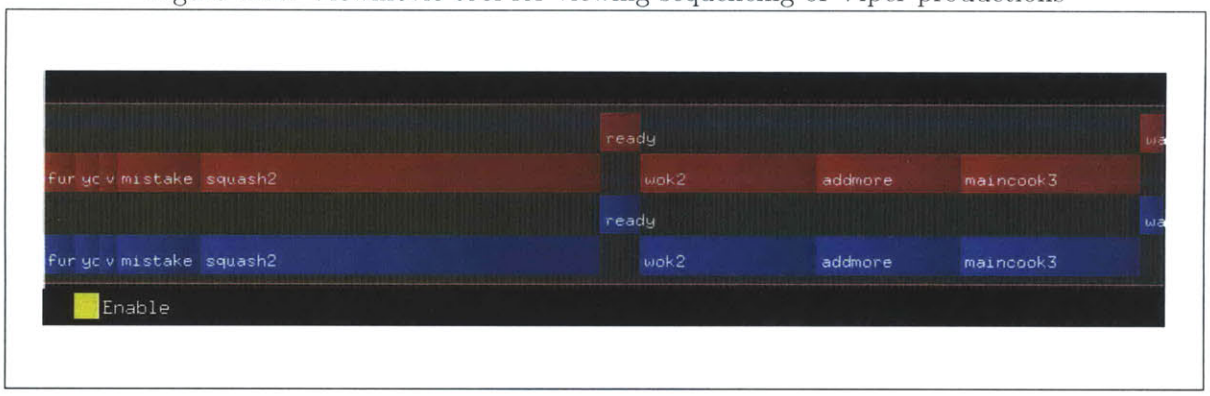
These steps (editing rules) were implemented in Isis using special Viper functions, which allow the content creator to selectively choose clips with certain types of annotations etc.

Figure 4.11: Affector Tool for viewing behavior




The diagram shows the use of a Viper tool called Affector, made for checking the behavior of a production. It is used to choose values for the factors. Another tool, called Viewmovie (in the next diagram) shows the sequencing of the clips on the time line for that particular set of values that the factors take. The red segments are video clips and the blue segments correspond to audio clips. Another tool, Playmovie, can be used to view the final production for a given set of factor values. The entire production consisted of 43 clips, with more than an hour of video footage.

Figure 4.12: Viewmovie tool for viewing sequencing of Viper productions



# Chapter 5

# Evaluation

## 5.1 The system and its components

The system was implemented successfully, and tests were performed to analyse the various factors that affected its performance. One factor that varied, and could have been improved upon, was the success of recognition of the ingredients. Using the techniques involved, there were eight different options during implementation of the recognition engine. These eight options arose due to three factors that could be varied during the training implementation, each factor having two possible states. They are:

1. *Matching to individual image mixture models vs. Matching to hierarchical mixture models*
   Matching to individual image mixture models may be more accurate if the training set captures all positions, forms and orientations of the ingredient, but would take longer computationally than a combined hierarchical.

2. *Training on complete images from the camera vs. Training on cropped images of the ingredient, extracted from the background using the separation technique mentioned in Section 3.3.*
   Complete images would add unnecessary noise to the density mixtures, while cropped images suffer the disadvantages of arbitrary sizes. Also, the object separation is not robust when shadows come into play, and this could cause problems during the matching stage.

3. *Training on full-sized images vs. Training on half-sized images.*
   Training on full-sized images would seem more accurate, but computationally intensive. Surprisingly though, the results in table 5.1 indicate that halving the size of the images,

not only speeds up the identification phase, but also gives better recognition. This behavior can be attributed to the loss of specular (shiny spots from lighting etc) and textural noise (from the mottled background of the table top) from smaller images, while retaining the essential shape and texture of the ingredients themselves. Indeed, there is an optimum size at which this is effective and any attempts to perform the matching at image resolutions less than 160x120 turned out to give arbitrary results, and consistent recognition rates were not observed.

Table 5.1: Choosing the right recognition technique

| Database Type | Accuracy |
|---|---|
| Full sized images: 320x240 | |
| Individual Images | 33% |
| Hierarchical Images | 33% |
| Ingredient-split Images | 52% |
| Hierarchical Ingredient-split Images | 52% |
| Half sized images: 160x120 | |
| Individual Images | 71% |
| Hierarchical Images | 79% |
| Ingredient-split Images | 79% |
| Hierarchical Ingredient-split Images | 79% |

We see from the table, that half-size images were better for recognition, due to the factors mentioned above. Also, hierarchical images performed as well or better than individual images, and their use is also justified because they are more scalable and less computationally expensive. Cropped images yielded better accuracy than full images, as predicted. Thus, the final system was trained using hierarchical models from half-sized images, with ingredients cropped from the background, and the same process was used to do the matching. The final system had approximately 80% accuracy, as long as lighting conditions were constant and different ingredients were kept sufficiently distant from each other, so that one would not cast a shadow on, or merge into the image of, the other.

Inconsistencies in recognition were observed when items of the same ingredient that looked sufficiently different from the training images were used, or when there were notable changes in lighting. Some of the future work should be aimed at training the system with a wide collection

of ingredients, and many versions of the same ingredient. This would allow the hierarchical models to capture the essential aspect of each ingredient, and make the system more robust to changes in appearance and lighting.

The processor time taken for computing the mixture models and performing the matching was also computed for full-size and half-size images. The computation time for full-size images was 28 seconds and that for half-size images was approximately 7 seconds. This reduction can be explained as the area decreases by a factor of four, causing a speed-up by an equivalent amount. In addition, it was necessary to return results within a few seconds after the user queries on an ingredient, and seven seconds was not acceptable. As mentioned in [1], the image is analyzed in blocks of 8 pixels by 8 pixels, with an overlap of 4 pixels in each step. By removing this overlap, a fourfold increase in speed was achieved. The overlap causes each pixel to be analyzed four times. a good feature for increasing the accuracy of matching, but given the requirements of responsiveness, a trade-off had to be achieved. The final search times were less than two seconds, and did not show significant drops in recognition accuracy.

## 5.2 Qualitative Survey

An evaluation of the system was performed by providing eleven users with a questionnaire. The questionnaire had four sections, with four to five questions in each section. The respondents included people from ages ranging from twenty to fifty, both male and female, with varying cooking abilities and interest in computers. Six respondents were asked to answer the questions before seeing the system, while five others were asked the same set of questions after looking at Minerva. The questionnaire, and answers to it, are included in appendix A.

**Statistical Significance** Due the small number of respondents to the data, a complete statistical analysis of the evaluation is not feasible. On the other hand, some minimum analysis of the data could be done, including testing the responses for statistical significance (which yields the confidence levels for making inferences from the responses).

When collecting data about a particular observation, we would like to know the trustworthiness of the data. Statistical significance approaches to calculate this trustworthiness work by assuming an initial hypothesis, called the "null hypothesis" to be true. The observed data is then analyzed, and if its calculated probability is seen to be below a certain cutoff probability, the null hypothesis is rejected, and an alternative hypothesis is accepted. The cutoff probability is chosen so that the chance of making both Type I errors (rejecting the null hypothesis when it is

actually true) and Type II errors (accepting the null hypothesis when it is actually false) is low. This is a trade-off since the probability of Type I errors and Type II errors are inversely related.

In the questions that we have chosen for this evaluation, there is no prior information that we know to decide a special hypothesis about the responses. Therefore, the null hypothesis is taken as the data being uniformly distributed (random). If we find that the probability of the results for a question are very low (below a cutoff), we reject this hypothesis, and say that the data is non-random, or statistically significant. In fact, we would prefer that the null hypothesis be rejected, as this would mean a high chance for the data to have some meaning or pattern, rather than just being random. In this questionnaire, instead of choosing a cutoff, we will simply give values for (1-p), where p is the probability of that response, as the statistical significance of that result. This technique is commonly used to provide a rough estimate of the relevance of the data. More information about these testing techniques is provided in [29]. Appendix A also includes values for the statistical significance values calculated for all the responses. In addition to this, comparative analysis between different test groups can be used to make inferences.

The following are simple inferences that can be made from the questionnaire, in spite of the limitations due to the low number of sample responses. The responses that have a statistical significance of more than 90% are used to make inferences while those above 80% are used to identify possible trends that could be tested with further evaluations. The results from the questionnaires indicate the following:

- *Overall impressions*: In general, most users were satisfied with the system. Those who had seen the system were enthusiastic about its applications, and said they would use it regularly (5) or on and off (6). Users who had not seen the system liked the idea of it from the description. A majority believed it would widen their options, be useful in giving them health information, and that they would like the suggestions. Also, people had a positive view of the interface and its uniqueness, but the low confidence level in these responses restricts us from making further inferences from the data. One respondent, however, said that they would prefer hyperlinks in the recipe, even if it would make it look more like a traditional computer system.

- A fair number of users (5/11) responded that they would like to use such a system "regularly". while the rest (6/11) said they would use it "on and off". There were no users who responded that they would "never" use the system. Minerva, as an entire system, therefore achieves its goals of creating a useful and unobtrusive cooking assistant and recipe retrieval system to a fair degree. It is interesting to note that two of the respondents who would

not use it regularly also said that they were "critical" about the applicability of computers in the home environment. One reason for people being wary of computers could be due to their inherent distrust which was likely created by prior experience with similar applications, but any concrete inferences have to be backed up by further surveys and evaluations. It would indeed be useful, in a future version of Minerva, to make it more comfortable to use without being obtrusive, in an effort to address these possible biases.

- Opinion over the health information portrayed by Minerva, and whether the health content of certain recipes should affect Minerva's behavior was divided, but as one respondent mentioned, "Information should always be displayed, but control over the action to be taken regarding this information should be with the user". The task to be addressed, in this scenario, is to give users control without destroying their involvement in the actual activity that they are involved in.

- A majority of the respondents (10/11) believed that suggestions from the computer on what to cook, would widen the set of choices that they could make. This is heartening, and indicates that suggestions are welcome, but their relevance to the situation is important. Increasing the accuracy of recognition would definitely increase the relevance of the suggestions, and work could be done towards taking the nature of the objects to be recognized to tailor the recognition algorithm.

- A few respondents mentioned that they would prefer a speech interface over a touch-screen interface. This is indeed a viable alternative, and frees the user's hands to continue with their activity (cooking in this case). Before exploring this option however, techniques for discriminating between the user's voice commands and background noise must be perfected, as well as making sure that the sound is not masked out by the audio from the cooking show or television program being presented.

The **'Student's T-test** is a statistical test [26] for judging the significance of the difference in means between two sets of sample data. Even though the number of samples was not large enough to render the analysis complete, the results from the test can be used to reinforce inferences made using other techniques. When the test is applied to data from two different samples, along with a hypothesis that is made about the relation between the populations represented by the two different samples, it yields a number for the probability that the hypothesis is an accurate description of the population.

The test was used to get a confidence level for the hypothesis that people who saw the system liked the idea of the system more and were enthusiastic about such applications than those who

didn't. If this hypothesis is true, it would give an indication that seeing the system had a positive impact on its users.

The test was done in the following stages:

- A subset of the evaluation questions was chosen for analysis. This set was based on the questions that reflected the respondents view of the advantages and friendliness of the Minerva system and its interface.

- For each of these questions, numbers were assigned to each of the options in the answers. These numbers were designed so that responses that indicated that the users liked the system and had a positive view of it, had higher values.

- The responses were split into two parts – responses from those who had seen the system and from those who hadn't. For each of these sets of responses, the answers to the questions were analyzed, and the numbers that weighted each option were summed up. Thus, for each respondent, one number was be obtained that indicated how much the user liked the system. On comparing these values between the people who had seen the system, and those who hadn't, one could get an estimate of whether seeing the system increased a user's appreciation of the interface and its goals.

- Once these numbers (indicated as "goodness values" in the table) are obtained, the mean and variance of the two sets are calculated. The t-value (as defined in the T-test) is then calculated using the equation shown in the figure.

  Once the t-value is obtained, a probability value, representing the confidence level in the hypothesis can be looked up in standard statistical tables using this t-value. A table look-up script [27] was used to calculate the values shown in the table. Values for sample set A correspond to the group that had not seen the system, and set B to the group that had seen it.

Figure 5.1: The 't' Statistic

$$t = \frac{\bar{x}_A - \bar{x}_B}{\sqrt{s_A^2/n_A + s_B^2/n_B}}$$

where $\bar{x}_A$ and $\bar{x}_B$ are the sample means of A and B, $s_A^2$ and $s_B^2$ are the sample variances of A and B, and $n_A$ and $n_B$ are the sample sizes of A and B.

| Parameter | Sample set A | Sample set B |
|---|---|---|
| "Goodness Values" from each questionnaire | | |
| Respondent 1 | 6.0 | 7.5 |
| Respondent 2 | 6.5 | 7.5 |
| Respondent 3 | 7.75 | 9.5 |
| Respondent 4 | 5.5 | 8.0 |
| Respondent 5 | 5.5 | 5.5 |
| Respondent 6 | 5.0 | — |
| Mean | 6.042 | 7.600 |
| Variance (square of the standard deviation) | 1.64 | 0.8 |
| Number of responses | 6 | 5 |

From the above values, the t-value was obtained to be **2.295** and the p-value corresponding to this was **0.0276**. This indicates that we can believe our hypothesis that viewing the system increases a user's positive view of its usefulness and applicability with a confidence level of **97.24%**. This is enough to what would traditionally be qualified statistically significant (95%) and re-inforces the views from the evaluations that Minerva was well-received by users, and that their view of its usefulness and usability increased after watching it in action.

**Further evaluations** A key feature in systems that address usability issues is, as seen above, the evaluation of its usage, features and interface. The following steps could be taken to further improve the relevance of the feedback obtained about Minerva, and use it to enhance the system.

- *More samples*: There is a definite need to have a larger set of evaluations so that accurate and useful inferences can be drawn from the data. While overall trends are hinted at, when using small samples, they may not be representative of the whole population. It is also necessary to have a diverse set of people to answer the questions – diversity in terms of age, gender, likes and dislikes, profession, nationality and culture etc.

- *Better Questions*: Designing questionnaires is a difficult task and requires keen insights into the type of questions that can result in useful responses – especially necessary are those that can discriminate the trends in the population and can be useful in analyzing correlations between the different topics. For example, a well-designed questionnaire will not only be able to extract information about the likes and dislikes of the users, but also how their preferences are dependent on other factors like location, age etc.

- *Better Techniques*: Sensible and clever techniques for analyzing the data can result in good inferences that may not be initially apparent. The responses must also be tested for significance and reliability using proper statistical techniques (split-halves method etc.).

# Chapter 6

# Conclusions and Future Work

## 6.1 Lessons Learned

While creating and using the Minerva system, there are a lot of instructive points about user interfaces and system design that can be noted. The importance of user control over his/her information, preferences and system behavior is very high, and a trade-off must be achieved between how much control is given to the users and how much is gleaned from their actions. Indeed, a system that gathers all information required for its functions from the user's actions would be totally easy to use, but at the same time, it would cause the user to suspect issues of control and privacy of information, and he/she could also be annoyed if unwarranted suggestions are brought up.

It is also necessary for a system to communicate effectively between the modules, while retaining the ability to use multiple techniques for doing the same step. This is achieved by establishing a set of protocols for communication between the modules. In the case of Minerva, all processes that are invoked by the Isis module write their output to a disk file that is later read by the parent process. Thus, if a new recognition engine based on a different technology is desired, the only requirements would be that it accepts an image input from a file, and returns the ingredient output to a file.

Extensibility of a system depends upon proper modular design of the internals. As mentioned earlier, since the display in Minerva was split into multiple screens that executed transitions between them when a specific action occurs, it was very easy to add new screens or remove them. Indeed, the screen to display Viper movies was added at the very end, but made to look exactly like a screen displaying a normal movie. Once the functionality of all the buttons was written down and the transitions were set, the system is ready to use the new screen.

Finally, the importance of context-awareness of a system is noticed only when people actually use the system. Thus, each small feature, like ordering the recipes in order of preference, choosing movies based on the preference, and even changing the presentation based on the user preference, contributes to the entire feel of the system. Users are more willing to use the system, and feel that the information is relevant to them when this is done. The qualitative evaluations are thus an important part of the iterative system design process.

## 6.2 Drawbacks

The Minerva system was evidently not perfect, and suffered a few drawbacks. They have been outlined below:

- The recognition system was not perfect, and a better system should be plugged in if Minerva is to be deployed for use in a kitchen. On the other hand, using the current system helped us understand the varying applications to which the image similarity functions in [1] could be applied.
- The touch-screen has a small delay in response due to queuing of data from the hardware. Instead of using Isis macros to read the serial port, it would be useful to directly use X windows drivers for the touch-screen.
- The SQL database queries are performed in a hard-coded fashion. It would be useful if the system learned the user's preferences during use and became better at retrieving relevant data about the recipes.
- The evaluation was not enough to capture the details of what users felt for the system. With a larger set of samples and thorough statistical analysis of the data received, many insights about the usefulness of the interface, and the system as a whole can be obtained.

## 6.3 Future Work

Minerva, as a cooking assistant, was noticed to be successful in achieving most of its goals of being a useful and unobtrusive cooking assistant with an interface that blends into the work space that it is situated in, and as a research tool, has offered new insights into object recognition, context-sensitive user interfaces, and extensible system design. Minerva could be extended to make a fully functional system that uses an extensive database with thousands of recipes, and a

wide recognition of ingredients. Indeed, it would be useful to extract the recipes and ingredient images (for training the recognition system) from the Internet, though this would require that the recipes and ingredient images are tagged with sufficient meta-data to indicate their content. In addition, directly changing the user's preferences through the touch-screen interface would be a useful feature.

It would also be preferable to make the system have more information about the user's instantaneous activity the kitchen appliances and other people in the room, so that it can respond in a more sensitive and useful manner. As an example, it could keep track of how the cooking is progressing, whether the kettle is boiling over, whether the user is interested in the cooking show etc. and correspondingly modify its behavior. Further insights can be obtained by creating and analyzing more evaluations. Some notes about further evaluations are mentioned in section 5.2.

Another feature, that inspired the original work in this direction, is to make connections between groups of people who are cooking the particular dish. This would not only liven up the scenario, but would also leverage the advantages of connectivity that computers offer us, in addition to their number-crunching powers. In conclusion, Minerva is a useful system that can have a large number of features added to it, and they will each be useful when incorporated into the system in a modular and user-friendly manner.

# Appendix A

# Qualitative Evaluation Questionnaire

The questionnaire that was provided for the evaluation, along with the answers from respondents, is given in this section. Only collective results of the answers are provided. Analysis of these results is presented in Chapter 5.

The background information about the respondents is given below.

1. *No. of respondents*: 11
2. *Age Range*: 20 to 50
3. *Gender*: 8 men, 3 Women
4. *Number of Nationalities*: 5

**The Basics**

This table shows the basic information that was collected of the participants. From the table below, it is apparent that their basic interest and experience levels in the fields that Minerva tackles were fairly varied (just as their age, gender and cooking preferences were). Thus, this gives a good range of answers. Even though the number of respondents is limited and large scale observations cannot be made, the statistical significance of each answer (the confidence level at which once can trust the responses) is given alongside. The number of responses for each option is also given next to the option.

| Questions | Option 1 | Option 2 | Option 3 | Stat. Sig. |
|---|---|---|---|---|
| Culinary Preference | Non-Vegetarian (7) | Vegetarian (4) | Vegan (0) | 99.8% |
| Rate your level of interest in computers | Low (0) | Medium (3) | High (8) | 99.9% |
| Rate your level of knowledge in User Interfaces and their Design | Low (2) | Medium (6) | High (3) | 97.5% |
| Rate your level of expertise in cooking | Low (5) | Medium (6) | High (0) | 99.7% |
| Would you describe yourself as adventurous in your cooking? | Yes (5) | No (6) | | 77.4% |
| Which of these words better describes your attitude towards computing in the home environment? | Appreciative (8) | Critical (3) | | 91.9% |

**Kitchen Computing**:

The following questions were designed to get a general view of what the respondents knew and thought about computing in the kitchen. It is interesting to analyze their answers to other questions with respect to the answers they provide in this table and those in the previous table.

| Questions | Option 1 | Option 2 | Option 3 | Stat. Sig. |
|---|---|---|---|---|
| What is your opinion on the use of comp. in the kitchen – How prevalent will they be? | Not Prevalent (2) | Medium (3) | Very Prevalent (6) | 97.5% |
| Would you like having a computer in the kitchen? | Yes (10) | No (1) | | 99.5% |
| Would you like having a television in the kitchen? | Yes (8) | No (3) | | 91.9% |
| Would you like suggestions and hints from a computer while cooking ? | Yes (7) | No (4) | | 83.9% |
| Would you like the computer to automatically understand your actions and suggest stuff or would you prefer complete control? | Automatic (8) | Complete Control (3) | | 91.9% |

**The Interface**:

These questions about the interface tried to judge if the specific Minerva interface could have been made in any other way that was more appealing and to judge its user friendliness.

| Questions | Option 1 | Option 2 | Stat. Sig. |
|---|---|---|---|
| Does the use of a camera to recognize foodstuff appeal to you over specifying the ingredients yourself, if the camera has only 80% accuracy? | Yes (7) | No (4) | 83.9% |
| Does the use of a touch-screen monitor appeal to you over other forms of interaction (speech etc. )? | Yes (6) | No (5) | 77.4% |
| Would you prefer a video cooking show over a cookbook with a list of recipes? | Yes (7) | No (4) | 83.9% |

**The System**:

Final questions to judge the overall success of the system, with regard to certain other factors that might give a better insight into their responses follow.

| Questions | Option 1 | Option 2 | Option 3 | Stat. Sig. |
|---|---|---|---|---|
| Given a particular set of ingredients, would a suggestion of what to cook with them, widen or constrain the choices you believe you could make? | Widen (10) | Constrain (0) | Neither (1) | 99.5% |
| Do your food preferences vary over time or are they fairly constant? | Vary (3) | Fairly Constant (8) | | 91.9% |
| Would you prefer a system that has health information built into it to restrict certain food (e.g. no chocolate if you are overweight )? | Yes (7) | No (4) | | 83.9% |
| Would you use the system on a regular basis, sometimes, or never? | Regular (5) | On and Off (6) | Never (0) | 99.7% |

# Bibliography


[1] Vasconcelos, Nuno. "Bayesian Models for Visual Information Retrieval." ,Ph.D Thesis, MIT, June 2000.

[2] Vasconcelos, Nuno and Andrew Lippman. " A Probabilistic Architecture for Content-based Image Retrieval", short version in CVPR'00, South Carolina, 2000.

[3] J.Berger. "Statistical Decision Theory and Bayesian Analysis", Springer-Verlag, New York, 1997.

[4] Challapalli, Sailabala. "Modeling the Temporal Components of Video Structure", M.Eng. Thesis, MIT, June 2001.

[5] Toshifumi Arai, Kimiyoshi Machii, Soshiro Kuzunuki and Hiroshi Shojima. "InteractiveDESK: a computer-augmented desk which responds to operations on real objects", CHI 95, May 1995, pp. 141-142.

[6] Want, R., Fishkin, K., Gujar, A., and Harrison, B. "Bridging Physical and Virtual Worlds with Electronic Tags", CHI '99, pp. 370-377.

[7] Constantine K. Christakos. "Taking Advantage of Distributed Multicasted Video to Deliver and Manipulate Television", S.M. Thesis, MIT, August 2000.

[8] Ruud M. Bolle, Jonathan H. Connell, Norman Haas, Rakesh Mohan and Gabriel Taubin. "VeggieVision: A Produce Recognition System", IEEE Workshop on Applications of Computer Vision, 244-251, 1996.

[9] T. Selker and W. Burleson. "Context-aware Design and Interaction in Computer Systems", IBM Systems Journal, Vol 39, No.s 3&4, 2000.

[10] H. Lieberman and T. Selker. "Out of Context: Computer Systems That Adapt To, and Learn From, Context", IBM Systems Journal, Vol 39, No.s 3&4, 2000.

[11] Bill Schilit, Norman Adams, Roy Want. "Context-Aware Computing Applications", IEEE Workshop on Mobile Computing Systems and Applications, Dec 1994.

[12] Counter Intelligence Group, Media Laboratory, MIT. [http://www.media.mit.edu/ci/]

[13] CNN Sci-Tech Computing Online, September 1998. [http://www.cnn.com/TECH/computing/9809/30/japan.internet.fridge/index.html]

[14] Wendy Ju, Rebecca Hurwitz, Tilke Judd and Bonny Lee. "CounterActive: An Interactive Cookbook for the Kitchen Counter", Personal Information Architecture Group, Media Laboratory, MIT. [http://www.media.mit.edu/~wendyju/counteractive-final.pdf]

[15] Edison Thomaz Jr. and Florian Mueller. "ImpactTV: Controlling Media with Physical Objects", Human Computer Interaction International 2001.

[16] Stefan Agamanolis. "Isis: A Programming Language for Responsive Media", Media Laboratory, MIT. [http://isis.www.media.mit.edu/]

[17] V. Michael Bove, Jr. and Stefan Agamanolis. "Responsive Television," Proc. IBC 2000, 2000.

[18] A. Van Dam, J.D. Foley, S. K. Fiener, and J.F. Hughes. "Computer Graphics: Principles and Practice", Addison-Wesley, June 1990.

[19] "bttv2", A v4l2 driver for Bt8x8 video cards. [http://bttv-v4l2.sourceforge.net/]

[20] Video for Linux Two (v4l2). [http://www.thedirks.org/v4l2/]

[21] miniSQL 2.0. [http://www.hughes.com.au/]

[22] Joseph 'Jofish' Kaye. "White Paper on Counter Intelligence", MIT Media Labs, June 1999.

[23] Matthew K. Gray. "Infrastructure for an Intelligent Kitchen",Masters Thesis, Media Laboratory, MIT, June 1999.

[24] Nitin Sawhney, Sean Wheeler and Chris Schmandt. "Aware Community Portals: Shared Information Appliances for Transitional Spaces", Personal Technologies, 2000.

[25] Susan Westmoreland (Editor). "The Good Housekeeping Step by Step Cookbook", Hearst Books, 2000.

[26] A T-test Tutorial. See [http://www.uow.edu.au/science/biol/biol104/ttest/tt230.html]

[27] T-test Online. See [http://home.clara.net/sisa/t-test.htm]

[28] 'Student's T-test for Independent Samples[http://www.ruf.rice.edu/ bioslabs/tools/stats/ttest.html]

[29] William Mendenhall, Wackerly, D.D. and Richard L. Schaeffer, "Mathematical Statistics with Applications", 1990, Boston: PWS-Kent.